k20 = fo(t+ h/2.0, r+ (k1r k2pr = fpr(t+ h/2.0, r+ (k2r)) k2po = fpo(t+ h/2.0, r + (k2r)) k3r = fr(t+ h/2.0, r+ (k2r)) k3o = fo(t+ h/2.0, r + (k2r)) k3pr = fpr(t+ h/2.0, r + (k2r)) k3po = fpo(t+ h/2.0, r + (k2r))	*h)/2.0, o+ (k10*h)/2.0, 1r*h)/2.0, o+ (k10*h)/2.0 k1r*h)/2.0, o+ (k10*h)/2 *h)/2.0, o+ (k20*h)/2.0, r*h)/2.0, o+ (k20*h)/2.0	pr+ (k1pr*h)/2.0, po+ (k1pr) pr+ (k1pr*h)/2.0, po+ (k1pr) pr+ (k1pr*h)/2.0, po+ (k1pr) pr+ (k1pr*h)/2.0, po+ (k1pr) pr+ (k2pr*h)/2.0, po+ (k2pr) pr+ (k2pr*h)/2.0, po+ (k2pr) pr+ (k2pr*h)/2.0, po+ (k2pr) pr+ (k2pr*h)/2.0, po+ (k1pr) pr+ (k1pr) p	o*h)/2.0) 1po*h)/2.0) k1po*h)/2.0) o*h)/2.0) po*h)/2.0) 2po*h)/2.0)			
<pre>k40 = fo(t+ h, r + k3r*h, k4pr = fpr(t+ h, r+ k3r*h, k4po = fpo(t+ h, r + k3r*h)  r = r + (h/6.0)*(k1r + 2.0 o = o + (h/6.0)*(k1o + 2.0 pr = pr + (h/6.0)*(k1pr + 2.0 pr = pr + (h/6.</pre>	k2r*h)/2.0, o+ (k2o*h)/2  + k3o*h, pr+ k3pr*h, po+ o+ k3o*h, pr+ k3pr*h, po- o+ k3o*h, pr+ k3pr*h, po- o+ k3o*h, pr+ k3pr*h, po- , o+ k3o*h, pr+ k3pr*h, po- * k2r + 2.0* k3r + k4r)  * k2o + 2.0* k3o + k4o) 2.0* k2pr + 2.0* k3pr + bare	+ k3po*h) o+ k3po*h) po+ k3po*h) k4pr)				
<pre>r_list.append(r)   o_list.append(o*(180.0/mth   pr_list.append(pr)   po_list.append(po)   x_list.append(x)   y_list.append(x)   y_list.append(xe)   ye_list.append(ye)   xm_list.append(ym)   ym_list.append(ym)   t_list.append(t)  plt.plot(x_list, y_list)   plt.axis([-30000,50000,-2000,10]   plt.rcParams["figure.figsize"]   plt.title("Trajectory of the Soplt.xlabel("x")   plt.ylabel("y")   plt.plot(xe_list[-1],ye_list[-1]   plt.annotate("Earth",(xe_list[-1])   plt.annotate("Moon",(xm_list[-1])   Text(-349090.9074498725, -17053)</pre>	8000]) = (20, 10) atellite")  1], marker="o", markersize -1], ye_list[-1])) 1], marker="o", markersize ], ym_list[-1]))		atellite			
15000 12500 10000 > 7500 2500				rth		
Satellite launched toward Comes back to Earth Surface  # Define the functions:  def s13(t,r,o):     return np.sqrt(r**2 + r1**;  def s23(t,r,o):     return np.sqrt(r**2 + r2**;  def fr(t,r,o,pr,po):     return pr # u = 1	Ce: 2 - 2.0*r*r1* np.cos(o -		20000	30000 40000	50000	
<pre>def fpo(t,r,o,pr,po):     return - (((gm1)/(s13(t,r,o))) #Define the constants:  rm = 50 # m1/m2 w = 2.0* (np.pi/(27.0*24.0)) # r1 = (3.84e5)/(1+rm) r2 = (3.84e5*rm)/(1+rm) gm1 = w**2 * r2 * 14.7e10 gm2 = gm1/rm  # Define the initial constants  ti = 0 tf = 0.47 # hours h = 0.01 # hours n = int((tf-ti)/h)</pre>	o)))*(r*r1* np.sin(o - w	r1* np.cos(o - w*t))) - (( *t))) + (((gm2)/(s23(t,r,o)				
<pre>r0 = r1 - 6400 # (km) launched o0 = 47*np.pi/36 # 235 degree pr0 = 8059 # km/hr po0 = 65*np.pi/36 # 325 degree  # start process :  t = ti r = r0 o = o0 pr = pr0 po = po0 x = r0* np.cos(o0) y = r0* np.sin(o0) xe = r1* np.cos(w*ti) ye = r1* np.sin(w*ti) xm = -r2* np.cos(w*ti) ym = -r2* np.sin(w*ti)  r_list = [r0] o_list = [00] pr_list = [pr0] po_list = [pr0] t_list = [ti] x_list = [r* np.cos(o)]</pre>						
<pre>y_list = [r* np.sin(o)] xe_list = [r1* np.cos(w*t)] ye_list = [r1* np.sin(w*t)] xm_list = [-r2* np.cos(w*t)] ym_list = [-r2* np.sin(w*t)]  for i in range(n):  #print(t, r, o, pr, po, x, y, xe, y) k1r = fr(t, r, o, pr, po) k1o = fo(t, r, o, pr, po) k1pr = fpr(t, r, o, pr, po) k1pr = fpr(t, r, o, pr, po) k1pr = fpo(t, r, o, pr, po) k2r = fr(t+ h/2.0, r+ (k1r) k2o = fo(t+ h/2.0, r+ (k1r) k2pr = fpr(t+ h/2.0, r+ (k2r) k3r = fr(t+ h/2.0, r+ (k2r) k3o = fo(t+ h/2.0, r+ (k2r) k3r = fr(t+ h/2.0, r+ (k2r)</pre>	*h)/2.0, o+ (k10*h)/2.0, *h)/2.0, o+ (k10*h)/2.0, 1r*h)/2.0, o+ (k10*h)/2.0, k1r*h)/2.0, o+ (k10*h)/2.0, *h)/2.0, o+ (k20*h)/2.0, r*h)/2.0, o+ (k20*h)/2.0, 2r*h)/2.0, o+ (k20*h)/2.0	pr+ (k1pr*h)/2.0, po+ (k1pr) pr+ (k1pr*h)/2.0, po+ (k1pr) pr+ (k1pr*h)/2.0, po+ (k1pr) pr+ (k1pr*h)/2.0, po+ (k1pr) pr+ (k2pr*h)/2.0, po+ (k2pr) pr+ (k2p	o*h)/2.0) 1po*h)/2.0) k1po*h)/2.0) o*h)/2.0) po*h)/2.0) 2po*h)/2.0)			
<pre>k4r = fr(t+ h, r+ k3r*h, or k4o = fo(t+ h, r + k3r*h, k4pr = fpr(t+ h, r+ k3r*h, k4po = fpo(t+ h, r + k3r*h)  r = r + (h/6.0)*(k1r + 2.0 o = o + (h/6.0)*(k1o + 2.0 pr = pr + (h/6.0)*(k1pr + 1.0 po = po + (h/6.0)*(k1po + 1.0 t = t+h x = r* np.cos(o) y = r* np.sin(o) xe = r1* np.cos(w*t) ye = r1* np.sin(w*t) xm = -r2* np.cos(w*t) ym = -r2* np.sin(w*t)  r_list.append(r) o_list.append(o*(180.0/mth pr_list.append(po) x_list.append(y) xe_list.append(xe)</pre>	o+ k3o*h, pr+ k3pr*h, po- o+ k3o*h, pr+ k3pr*h, po- , o+ k3o*h, pr+ k3pr*h, pr- * k2r + 2.0* k3r + k4r) * k2o + 2.0* k3o + k4o) 2.0* k2pr + 2.0* k3pr + 1 2.0* k2po + 2.0* k3po + 1	+ k3po*h) o+ k3po*h) po+ k3po*h) k4pr)				
<pre>ye_list.append(ye) xm_list.append(xm) ym_list.append(ym) t_list.append(t)  plt.plot(x_list, y_list) plt.axis([-30000,50000,-5000, plt.rcParams["figure.figsize"] plt.title("Trajectory of the Soplt.xlabel("x") plt.ylabel("y") plt.ylabel("y") plt.plot(xe_list[-1],ye_list[-1]) plt.annotate("Earth",(xe_list[-1]) plt.annotate("Moon",(xm_list[-1])</pre>	= (20, 10) atellite")  1], marker="o", markersize -1], ye_list[-1]))  1], marker="o", markersize ], ym_list[-1]))		atellite			
2000						
> 0 -2000		E rth				
		0 10000 x	20000	30000 40000	50000	
-2000  -3000 -20000  Orbits around earth:  # Define the functions:  def s13(t,r,o):     return np.sqrt(r**2 + r1**;  def s23(t,r,o):     return np.sqrt(r**2 + r2**;  def fr(t,r,o,pr,po):     return pr # u = 1  def fo(t,r,o,pr,po):     return (po**2/(r**3)) - (((     def fpo(t,r,o,pr,po):         return - (((gm1)/(s13(t,r,t))));  #Define the constants:  rm = 50 # m1/m2 w = 2.0* (np.pi/(27.0*24.0)) # r1 = (3.84e5)/(1+rm) r2 = (3.84e5*rm)/(1+rm) gm1 = w**2 * r2 * 14.7e10 gm2 = gm1/rm	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o - (gm1)/(s13(t,r,o)))*(r - o)))*(r*r1* np.sin(o - w	0 10000 x	(gm2)/(s23(t,r,o)))*(r + r2	2* np.cos(o - w*t)))	50000	
-2000  -30000 -20000  Orbits around earth:  # Define the functions:  def s13(t,r,0):     return np.sqrt(r**2 + r1**;  def s23(t,r,0):     return np.sqrt(r**2 + r2**;  def fr(t,r,0,pr,p0):     return pr # u = 1  def fo(t,r,0,pr,p0):     return (po**2/(r**3)) - (((     def fpr(t,r,0,pr,p0):         return - (((gm1)/(s13(t,r,t))))  #Define the constants:  rm = 50 # m1/m2 w = 2.0* (np.pi/(27.0*24.0)) #. r1 = (3.84e5)/(1+rm) r2 = (3.84e5*rm)/(1+rm) gm1 = w**2 * r2 * 14.7e10 gm2 = gm1/rm  # Define the initial constants  ti = 0 tf = 1.5 # hours h = 0.01 # hours n = int((tf-ti)/h) r0 = r1 - 6400 # (km) launched 00 = 47*np.pi/36 # 235 degree  # start process:  t = ti r = r0 0 = 00 pr = pr0 po = po0 x = r0* np.sin(00) y = r0* np.sin(00) y = r0* np.sin(00) y = r1* np.cos(w*ti) y = r1* np.cos(w*ti) y = r1* np.sin(w*ti)	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o -  (gm1)/(s13(t,r,o)))*(r -  o)))*(r*r1* np.sin(o - with the company of the co	0 10000 x w*t))**3 w*t))**3	(gm2)/(s23(t,r,o)))*(r + r2	2* np.cos(o - w*t)))	50000	
-2000  -30000 -20000  Orbits around earth:  # Define the functions:  def \$13(t,r,0):     return np.sqrt(r**2 + r1**;  def \$23(t,r,0):     return np.sqrt(r**2 + r2**;  def fr(t,r,0,pr,p0):     return pr # u = 1  def fo(t,r,0,pr,p0):     return po/(r**2)  def fpr(t,r,0,pr,p0):     return (po**2/(r**3)) - (((     def fpo(t,r,0,pr,p0):         return - (((gm1)/(s13(t,r,t))))  #Define the constants:  rm = 50 # m1/m2 w = 2.0* (np.p1/(27.0*24.0)) # r1 = (3.84e5)/(1+rm) r2 = (3.84e5*rm)/(1+rm) gm1 = w**2 * r2 * 14.7e10 gm2 = gm1/rm  # Define the initial constants  ti = 0  tf = 1.5 # hours h = 0.01 # hours n = int((tf-ti)/h) r0 = r1 - 6400 # (km) launched o0 = 47*np.pi/36 # 235 degree pr0 = 20559 # km/hr po0 = 65*np.pi/36 # 325 degree  # start process:  t = ti r = r0 o = o0 pr = pr0 po = po0 x = r0* np.cos(o0) y = r0* np.sin(o0) xe = r1* np.cos(w*ti) ym = -r2* np.cos(w*ti) ym = -r2* np.sin(w*ti)  r_list = [r0] o_list = [p00] pr_list = [p00] pr_list = [p00] plist = [ti] x_list = [ti] x_list = [r* np.cos(o)]	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o -  (gm1)/(s13(t,r,o)))*(r -  o)))*(r*r1* np.sin(o - w  rad/hour  :  towards moons	0 10000 x w*t))**3 w*t))**3	(gm2)/(s23(t,r,o)))*(r + r2	2* np.cos(o - w*t)))	50000	
-2000  -30000 -20000  Drbits around earth:  # Define the functions:  def s13(t,r,o):     return np.sqrt(r**2 + r1**;  def s23(t,r,o):     return pr  # u = 1  def fo(t,r,o,pr,po):     return pr  # u = 1  def fo(t,r,o,pr,po):     return po/(r**2)  def fpr(t,r,o,pr,po):     return po/(r**2)  def fpo(t,r,o,pr,po):     return - (((gm1)/(s13(t,r,w)))  #Define the constants:  rm = 50 # m1/m2 w = 2.0* (np.p1/(27.0*24.0)) # r1 = (3.84e5)/(1+rm) gm1 = w**2 * r2 * 14.7e10 gm2 = gm1/rm  # Define the initial constants  ti = 0  tf = 1.5 # hours h = 0.01 # hours n = int((tf-ti)/h) r0 = r1 - 6400 # (km) launched 00 = 47*np.pi/36 # 235 degree pr0 = 20559 # km/hr po0 = 65*np.pi/36 # 325 degree pr0 = 20559 # km/hr po0 = 65*np.pi/36 # 325 degree  # start process:  t = ti r = r0 0 = 00 pr = pr0 po = po0 x = r0* np.cos(w*ti) ye = r1* np.cos(w*ti)	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o -  (gm1)/(s13(t,r,o)))*(r -  0)))*(r*r1* np.sin(o - w  rad/hour  :  towards moons  towards moons  towards h, pr + k3pr + p. p. o. the shape h, pr + k3pr + p. p. the shape h,	pr+ (k1pr*h)/2.0, po+ (k1ppr+ (k1pr*h)/2.0, po+ (k1pp0, pr+ (k1pr*h)/2.0, po+ (k2p, pr+ (k2pr*h)/2.0, po+ (k2p), pr+ (k2p),	(gm2)/(s23(t,r,o)))*(r + r2 ))*(r*r2* np.sin(o - w*t))) po*h)/2.0) po*h)/2.0) po*h)/2.0) po*h)/2.0) po*h)/2.0) pop*h)/2.0)	2* np.cos(o - w*t)))	50000	
-2000  -2000  -2000  Drbits around earth:  # Define the functions:  def s13(t,r,o):     return np.sqrt(r**2 + r1**.  def s23(t,r,o):     return pr.sqrt(r**2 + r2**.  def f23(t,r,o,pr,po):     return pr.sqrt(r**2 + r2**.  def ff(t,r,o,pr,po):     return po/(r**2)  def fp(t,r,o,pr,po):     return (p(o**2/(r**3)) - ((  def fp(t,r,o,pr,po):     return ((((gm1)/(s13(t,r,f))))  ### 1 (3.8465/(1)*rm)  r2 = (3.8465*m)/(1*rm)  gm1 = w**2 * r2 * 14.7e10  gm2 = gm1/rm  # Define the initial constants  ti = 0  tf = 1.5 # hours     t = 0.1 # hours     t = 0.0 1 # hours     n = 1nt((tf-t1)/h)     r0 = r1 - 6400 # (km) launched     00 = 47*np.p1/36 # 235 degree     pr0 = 20559 # km/hr     po0 = 65*np.p1/36 # 325 degree  # start process:  t = ti     r = r0     0 = 00     pr = pr0     po = po0     x = r0* np.cos(00)     y = r0* np.sin(00)     x = r0* np.sin(00)     x = r1* np.cos(w*ti)     ye = r1* np.cos(w*ti)     ye = r1* np.cos(w*ti)     ye = r1* np.sin(w*ti)     x = -r2* np.cos(w*ti)     ye = r1* np.cos(w	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o -  (gm1)/(s13(t,r,o)))*(r -  0)))*(r*r1* np.sin(o - w  rad/hour  :  towards moons  towards moons  towards moons  towards moons  ye, xm, ym)  *h)/2.0, o+ (k10*h)/2.0, o+ (k10*h)/2.0, o+ (k20*h)/2.0, o+ (k10*h)/2.0, o+ (k20*h)/2.0, o+ (k20*	pr+ (k1pr*h)/2.0, po+ (k1ppr+ (k1pr*h)/2.0, po+ (k1pp0, pr+ (k1pr*h)/2.0, po+ (k2p, pr+ (k2pr*h)/2.0, po+ (k2p), pr+ (k2p),	(gm2)/(s23(t,r,o)))*(r + r2 ))*(r*r2* np.sin(o - w*t))) po*h)/2.0) po*h)/2.0) po*h)/2.0) po*h)/2.0) po*h)/2.0) pop*h)/2.0)	2* np.cos(o - w*t)))		
-2000  -3000 -2000  Drbits around earth:  # Define the functions:  def s13(t,r,0):     return np.sqrt(r**2 + r1**;  def s23(t,r,0):     return pr.sqrt(r**2 + r2**;  def fr(t,r,0,pr,po):     return pr # u = 1  def fo(t,r,0,pr,po):     return po/(r**2)  def fp(t,r,0,pr,po):     return (p0**2/(r**3)) - ((  def fpo(t,r,0,pr,po):     return (p0**2/(r**3)) - ((  def fpo(t,r,0,pr,po,x,y,ze,d,x,ze,d,ze,d,ze,d,ze,d,ze,d,ze,d,ze	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o -  (gm1)/(s13(t,r,o)))*(r -  o)))*(r*r1* np.sin(o - w  rad/hour  :  :  towards moons  towards moons  towards moons  towards moons  towards moons  selection of the	pr+ (k1pr*h)/2.0, po+ (k1pp+ (k1pr*h)/2.0, po+ (k1pp+ (k1pr*h)/2.0, po+ (k2p0, po+ (k2p1, po+ (k2p1	(gm2)/(s23(t,r,o)))*(r + r2 ))*(r*r2* np.sin(o - w*t)); tpo*h/2.0) k1po*h/2.0) po*h/2.0) po*h/2.0) k2po*h/2.0) k2po*h/2.0)	2* np.cos(o - w*t)))		
-2000  -2	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o -  (gm1)/(s13(t,r,o)))*(r -  o)))*(r*r1* np.sin(o - w  rad/hour  :  :  towards moons  towards moons  towards moons  towards moons  towards moons  selection of the	pr+ (klpr*h)/2.0, po+ (klp pr+ (klpr*h)/2.0, po+ (klp 0, pr+ (klpr*h)/2.0, po+ (klp pr+ (k2pr*h)/2.0, po+ (klp pr+ (k2pr*h)/2.0, po+ (klp pr+ (k2pr*h)/2.0, po+ (klp 0, pr+ (k2pr*h)/2.0, po+ (klp 0, pr+ (klpr*h)/2.0, po+ (klpr*h)/2.0, po+ (klpr*h)/2.0, po+ (klpr*h)/2.0, po+ (klpr*h)/2.0, po+ (klpr*h)/2.0,	(gm2)/(s23(t,r,o)))*(r + r2 ))*(r*r2* np.sin(o - w*t)); tpo*h/2.0) k1po*h/2.0) po*h/2.0) po*h/2.0) k2po*h/2.0) k2po*h/2.0)	2* np.cos(o - w*t)))	50000	
	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o - 2 + 2.0*r*r2* np.cos(o - 4 + 2.0*r*r2* np.cos(o - 4 + 2.0*r*r2* np.sin(o - 4 + 2.0*r*r2* np.cos(o - 4 + 2.0*r*r2* np.cos(	pr+ (k1pr*h)/2.0, po+ (k1pp+ (k1pr*h)/2.0, po+ (k2pp+ (k2pr*h)/2.0, po+ (k2p, pr+ (k2pr*h)/2.0, po+ (k2pr*h)/2.0, po+ (k2p, pr+ (k2pr*h)/2.0, po+ (k2p, pr+ (k2pr*h)/2.0, po+ (k2p, pr+ (k2pr*h)/2.0, po+ (k2p	(gm2)/(s23(t,r,o)))*(r + r2*) (y*r2* np.sin(o - w*t)); (p*p*)/2.0) (p*p*)/2.0) (p*p*)/2.0) (p*p*)/2.0) (p*p*)/2.0) (p*p*)/2.0) (p*p*)/2.0) (p*p*)/2.0)	2* np.cos(o - w*t))) )  30000 40000		
### Orbits around earth:  ### Define the functions:  def si3(t,r,o):     return np.sqr(r*2 + r1**  def \$23(t,r,o):     return np.sqr(r*2 + r2**,  def fr(t,r,o,pr,po):     return pr # u = 1  def fo(t,r,o,pr,po):     return pr (r*2)  def fp(t,r,o,pr,po):     return (po*2/(r*3)) - ((  def fp(t,r,o,pr,po):     return (po*2/(r*3)) - ((  def fp(t,r,o,pr,po):     return - (((gm1)/(s13(t,r,r))) + ((gm1)/(s13(t,r,r)))	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o - 2 + 2.0*r*r2* np.cos(o - 4 + 2.0* moons  (gm1)/(s13(t,r,o)))*(r - 5 + 2.0* moons  (gm1)/(s13(t,r,o)))*(r - 6 + 2.0* moons  **rad/hour*  **towards moons  **towards moons  **towards moons  (gm1)/(s13(t,r,o)))*(r - 6 + 2.0* moons  **towards moons	pr+ (klpr*h)/2.0, po+ (klp pr+ (klpr*h)/2.0, po+ (klp pr+ (klpr*h)/2.0, po+ (klp pr+ (klpr*h)/2.0, po+ (klp 0, pr+ (k2pr*h)/2.0, po+ (klp 0, pr+ (k2pr*h)/2.0, po+ (klp 0, pr+ (k2pr*h)/2.0, po+ (klp 0, pr+ (k2pr*h)/2.0, po+ (klp k3po*h) po+ k3po*h) po+ k3po*h) po+ k3po*h) k4po)  Trajectory of the:	(gm2)/(s23(t,r,o)))*(r + r2) (pm2)/(s23(t,r,o)))*(r + r2) (pm2)/(s23(t,r,o	2* np.cos(o - w*t))) )		
	2 - 2.0*r*r1* np.cos(o - 2 + 2.0*r*r2* np.cos(o - 2 + 2.0*r*r2* np.cos(o - 4 + 2.0* moons  (gm1)/(s13(t,r,o)))*(r - 5 + 2.0* moons  (gm1)/(s13(t,r,o)))*(r - 6 + 2.0* moons  **rad/hour*  **towards moons  **towards moons  **towards moons  (gm1)/(s13(t,r,o)))*(r - 6 + 2.0* moons  **towards moons	pr+ (k1pr*h)/2.0, po+ (k1p pr+ (k1pr*h)/2.0, po+ (k1p 0, pr+ (k1pr*h)/2.0, po+ (k2p, pr+ (k2pr*h)/2.0, po+ (k2p, p	(gm2)/(s23(t,r,o)))*(r + r2) (pm2)/(s23(t,r,o)))*(r + r2) (pm2)/(s23(t,r,o	2* np.cos(o - w*t))) )		
### Joseph   John   Joh	2 - 2.8*r*r1* np.cos(o - 2 + 2.8*r*r2* np.cos(o - 2 + 2.8*r*r2* np.cos(o - 3 + 2.8*r*r2* np.cos(o - 4 + 2.8*r3* np.cos(o - 5 +	pr+ (kipr*h)/2.0, po+ (kipp*h)/2.0, po+ (kipp*h)/2.0, po+ (kipp*h)/2.0, po+ (kip, pr+ (kipr*h)/2.0, po+ (kip), pr+ (kipr*h)/2.0, po+ (ki	(gm2)/(s23(t,r,o)))*(r+r) (y*r*2* np.sin(o - w*t)) (y*n)/2.0) (ph/2.0)	2* np.cos(o - w*t))) )		
### Portion of the functions:  ### provided by the functions:  #### provided by the functions:  ### provided by the fun	2 - 2.0*r*r1* np.cos(0 - 2 + 2.0*r*r2* np.cos(0 - 2 + 2.0*r*r2* np.cos(0 - 2 + 2.0*r*r2* np.cos(0 - 3 + 2.0*r*r2* np.cos(0 - 4 + 2.0*r*r2* np.cos(0 - 2 + 2.0*r*r2* np.cos(0 - 4 + 2.0*r*r2* np.cos(0 - 2 + 2.0*r*r2* np.cos(0 - 4 + 2.0*r2* np.cos(0 - 4 + 2.0*	pr+ (klpr*h)/2.0, po+ (klpr*h)/2.0, po+ (klpr*h)/2.0, po+ (klp. pr+ (klpr*h)/2.0, po+ (klp. po+ (klp. po+ (klp. pr+ (klpr*h)/2.0, po+ (klp. po+ (klp	(gm2)/(s23(t,r,o)))*(r + r2*))*(r*r2* np.sin(o - w*t));  o*h)/2.0) w*h)/2.0) (pm*h)/2.0) (pp*h)/2.0) pp*h*h/2.0) pp*h*h/2.0) pp*h*h/2.0) pp*h*h/2.0)	2* np.cos(o - w*t))) )		
### ### ### ### ### ### ### ### ### ##	2 - 2.8*r*r1* np.cos(0 - 2 2 + 2.8*r*r2* np.cos(0 - 2 2 + 2.8*r*r2* np.cos(0 - 3 (gm1)/(s13(t,r,0)))*(r - 4 )))*(r*r1* np.sin(0 - w rad/hour  **rad/hour  **towards mons  **to	pr+ (kipr*h)/2.0, po+ (kipp*, rep*, kipr*h)/2.0, po+ (kipp*h)/2.0, p	(gm2)/(s23(t,r,o)))*(r + r2*))*(r*r2* np.sin(o - w*t));  o*h)/2.0) w*h)/2.0) (pm*h)/2.0) (pp*h)/2.0) pp*h*h/2.0) pp*h*h/2.0) pp*h*h/2.0) pp*h*h/2.0)	2* np.cos(o - w*t))) )		